REMARKS

Claims 1-16 were presented for examination and all have been rejected under 35 U.S.C. § 102. Claims 1, 4, 5, 6, 9 and 10 are being amended to claim in the alternative language. In view of the above amendment and following remarks, reconsideration of the claims is respectfully requested.

REJECTIONS UNDER 35 U.S.C. § 102(b) – Copperman (PMW)

In paragraphs 4 and 5 of the Final Office Action, rejection of claims 1-10 in the Office Action mailed December 29, 2003 was repeated, and rejection of claims 11-16 were added. In brief, claims 1-16 was rejected under 35 U.S.C. § 102 (b) as being anticipated by *Copperman et al*, "Poor Man's Watchpoint" (PMW).

Claim 1 is being amended to claim in the alternative language. As indicated in the Amendment ("Amendment") in response to the Office Action, claim 1 recites a block of original code and a block of instrumented code, and the block of original code includes an instrumentation breakpoint. The Amendment further recites '[b]ecause the instrumentation breakpoint is included in the block of original code, it is implied that the block of *original code is invoked for the instrumentation breakpoint to be reached*, then the block of instrumented code is generated" (Amendment, page 6, near the top, emphasis added). However, as the Final Office Action seemed not to appreciate this point, claim 1, as amended, recites "running the block of original code including the instrumentation breakpoint until a breakpoint is reached." Amended claim 1 further recites "determining whether the breakpoint is the instrumentation breakpoint; and, if so, then generating the block of instrumented code; and performing debugging functions on the block of instrumented code at the reached debugging breakpoint." In effect, in claim 1, the block of original code is run, a breakpoint is reached and if the breakpoint is the instrumentation breakpoint,

then the block of instrumented code is generated, and debugging is performed on this block of instrumented code when a debugging breakpoint is reached.

In contrast, PMW, during debugging, does not deal with the block of original code, but deals only with the instrumented code, i.e., the code with the patch of _do_watch. PMW does not teach, suggest, or make obvious providing an instrumentation breakpoint in the block of original code or generating the block of instrumented code if the reached breakpoint is the instrumentation breakpoint included in the original code as in claim 1. Further, as indicated in the Amendment "*PMW*'s debugger sees only the instrumented code, and there is no teaching of programming flow between the original code and the instrumented code" (Amendment, page 6, near the middle) as in claim 1. That is, once the original block of code is run, the instrumentation breakpoint in the block of original code is reached, then the block of instrumented code is generated and run until a debugging breakpoint is reached.

The Final Office Action asserted that PMW discloses providing an instrumentation breakpoint in the block of original code, and generating the block of instrumented code when the instrumentation breakpoint is reached because "a call to _do_watch is patched into the executable prior to each memory access" and "[t]he instrumentation breakpoint is what causes the _do_watch function to be patched." It is respectfully submitted that there is nothing in PMW to suggest that "the instrumentation breakpoint is what causes the _do_watch function to be patched." In fact, both the Office Action and the Final Office Action failed to correspond the claimed "instrumentation breakpoint" to an element of PMW. A general conclusion that "[t]he instrumentation breakpoint is what causes the _do_watch function to be patched" without particularly pointing to an element corresponding to the claimed *instrumentation breakpoint* is improper for a 35 U.S.C § 102 rejection as in this case.

The Final Office Action further asserted "[t]his [_do_watch] patch is generated into the executable, as stated on page 38, 'A post-loader is a program that inserts instrumentation code into executables,' thereby ultimately generating an executable with instrumented code patched in. As such, PWM discloses providing an instrumentation breakpoint in the block of original code, and generating the block of instrumented code when the instrumentation breakpoint is reached as claimed." It is respectfully submitted that the cited paragraph discloses inserting instrumentation code into executables, but does not teach generating the block of instrumented code *when* the instrumentation breakpoint is reached, or as currently claim, "running the block of original code including the instrumentation breakpoint until a breakpoint is reached; determining whether the breakpoint is the instrumentation breakpoint; and, if so, then generating the block of instrumented code." The Amendment has indicated "Applicants' claim 1 recites 'generating the block of instrumented code *when the instrumentation breakpoint is reached*' (emphasis added). Even though *PMW* discloses 'replacing each store and/or load instructions with an inline check . . .' (page 37, near the end), *PMW* fails to teach such replacement occurs *when the instrumentation breakpoint is reached.* For the sake of argument, if *PMW*'s call to _do_watch corresponded to Applicants' instrumentation breakpoint, then, to be parallel with Applicants' claim 1, the replacement of the store and/or load instructions with an inline check or call to a function, which the Examiner corresponded to Applicants' block of instrumented code, *must be generated when a call to _do_watch is reached.* However, there is no such teaching in *PMW*" (Amendment, page 7, first full paragraph). Such indication in the Amendment clearly distinguishes another element of claim 1 from PMW, and there was no response to such distinguishing in the Final Office Action.

Additionally, claim 1 includes two types of breakpoint, the instrumentation breakpoint and the debugging breakpoint, embodiments of which are illustrated in Applicants' Specification (page 5 and 6), but PMW discloses only one type of breakpoint.

Because claim 1 recites limitations patentably distinguished from PMW, claim 1 is patentable.

Claims 2-5 depend directly or indirectly from claim 1 and are therefore patentable for at least the same reasons as claim 1. Claims 2-5 are also patentable for their additional limitations as appropriate.

The Amendment indicated "claims 2 and 3 recite 'the step of providing comprises the step of replacing a fist instruction in the block of original code . . .,' which is not taught, suggested, or made obvious by *PMW*. The Examiner asserted that a call to _do_watch is patched into the executable prior to each memory access, and '[t]he patch replaces the code section before the memory access block of code.' However, Applicants respectfully submit that there is no teaching of such replacement in *PMW*. Code may be patched into another code section without replacing other code or instructions" (Amendment, page 7, second paragraph from the bottom). The Final Office Action asserted that "[t]he instrumentation breakpoint is what causes the patching, and it would inherently replace a first instruction." As indicated above, the general conclusion that "[t]he instrumentation breakpoint is what causes the patching" without particularly identifying what corresponds the claimed *instrumentation breakpoint* is improper. Further, the assertion that "it would *inherently* replace a first instruction" (emphasis adde) because "[n]ot replacing the instruction would cause the instruction to execute twice, once in the patched version, and once in the non-patched version, resulting in the extra execution of an instruction and an error in program operation" is also improper. As indicated in the Amendment, "[c]ode may be patched

into another code section without replacing other code or instructions," let alone replacing a *first* instruction. Further, a general conclusion of inherency is improper.

Because PMW does not disclose, the "instrumentation breakpoint" and the "first instruction" replaced by the instrumentation breakpoint, PMW cannot logically disclose "the first instruction comprises one or more instruction" (claim 3) or "restoring the first instruction." (claim 4).

Claims 6-10 recite limitations corresponding to claim 1-5, and are patentable for at least the same reasons as claim 1-5.

Claims 11-16, which depend from claims 1 or 6 are patentable for at least the same reasons as claims 1 or 6 from which claims 11-16 depend. Claims 11-16 are also patentable for their limitations.

Regarding claim 11 and 14, PMW does not disclose the first instruction replaced by the instrumentation breakpoint, and therefore cannot logically disclose that the first instruction being a second debugging breakpoint. The assertion in the Final Office Action that "the first instruction replaced by the instrumentation breakpoint *inherently* becomes a second debugging breakpoint" (emphasis added) is improper. For the rejection to stand, it must be shown that the prior art, e.g., PMW, discloses or suggests that the first instruction being a debugging breakpoint. A general conclusion of inherency is improper. The explanation "as every _do_watch call causes a breakpoint to be reached" has nothing to do with the claimed first instruction being a debugging breakpoint.

Regarding claim 12 and 15, the assertion in the Final Office Action that "the instrumentation breakpoint inherently becomes a second debugging breakpoint" (emphasis added) is improper. For the rejection to stand, it must be shown that the prior art, e.g., PMW, discloses or suggest that the instrumentation breakpoint being a debugging breakpoint. A general conclusion of inherency is improper. The

explanation "as every _do_watch call causes a breakpoint to be reached" has nothing to do with the claimed instrumentation breakpoint being replaced by a debugging breakpoint.

Regarding claim 13 and 16, the assertion that "an instrumentor would *inherently* have provided the instrumentation breakpoint" (emphasis added) is improper. For the rejection to stand, it must be shown that the prior art, e.g., PMW, discloses or suggest that an instrumentor provides the instrumentation breakpoint. A general conclusion of inherency is improper. PMW does not disclose that the instrumentor provides *information* related to the instrumentation breakpoint, either.

SUMMARY

In conclusion, it is respectfully submitted that pending claims 1-16 present subject matter that is patentable over the prior art of record, and therefore withdrawal of the rejection and reconsideration of the claims are respectfully requested.

Respectfully submitted,

Robert Hundt et al

Date: 8/16/04 By: _____

Tuan V. Ngo, Reg. No. 44,259
IP Administration
Legal Department, M/S 35
Hewlett-Packard Company
P. O. Box 272400
Fort Collins, CO 80527-2400
Phone (408) 447-8133
Fax (408) 447-0854